UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/656,097	09/05/2003	Bhushan Khaladkar	OI7035732001	9920

55498 7590 05/28/2009

ORACLE INTERNATIONAL CORPORATION
c/o VISTA IP LAW GROUP LLP
1885 LUNDY AVENUE
SUITE 108
San Jose, CA 95131

EXAMINER	
TSUI, WILSON W	
ART UNIT	**PAPER NUMBER**
2178	
MAIL DATE	**DELIVERY MODE**
05/28/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No. 10/656,097	Applicant(s) KHALADKAR ET AL.	
	Examiner WILSON TSUI	Art Unit 2178	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _13 March 2009_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-62_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-62_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _20090331_.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. This non-final action is in response to the RCE filed on: 03/13/09, the IDS filed on: 03/31/09.

2. Claims 1, 11, 18, 30 - 35 are amended. Claims 1-62 are pending. Claims 1, 11, 18, 30-35 are independent claims.

3. The following rejections are withdrawn, as necessitated by applicant's amendments:

- Claims 1, 2, 3, 4, 8, 10, 11, 12, 14, 21, 30, 31-33, 37, 38, 40, 42, 43, 45, 56, 60 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy in view Upton, IV in further view of Dreyband et al, in further view of Bornstein.
- Claims 5, 6, 7, 9, 13, 39, 41, 44, 46, 55, and 59 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Upton, IV, and in view of Dreyband et al and further in view of JAXB.
- Claims 15-17, 57, 58, 61, and 62 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Upton, IV, in view of Dreyband et al, in view of Bornstein, and further in view of Wan.
- Claims 18, 19, 20, 23, 24, 29, 34-36, 47, 48, and 50-54 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view Upton, IV, and further in view of Bornstein.
- Claims 22, 49, and 53 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Upton, IV, and further in view of JAXB.

- Claims 25, 26, 27, and 28 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Upton, IV, further in view of Bornstein, and further in view of Wan.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-4, 8, 10, 11, 12, 14, 18 – 21, 23, 24, 29, 30 – 38, 40, 42, 43, 45, 47, 48, 50 - 52, 54, 56, 60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy (US Patent: 6,662,342 B1, published: Dec. 9, 2003, filed: Dec. 13, 1999), in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), and further in view of Bhatt et al (US Patent: 6,799,184 B2, issued: Sep. 28, 2004, filed: Jan. 30, 2002, submitted in ids filed on 3/31/09).

With regards to claim 1, Marcy teaches a method comprising:

- *Receiving a schema for the XML data*, as similarly explained in the rejection for claim 18, and is rejected under the same rationale.

- *Storing the XML data in a database* (column 4, lines 1-29: whereas, XML data is stored in a document. The document is stored in computer memory/data-store/database).
- *Determining using a processor at least one access parameter for the element relative to a second element, in accordance with a schema, wherein the at least one access parameter allows the access procedure to have direct access to the element, … without progressive traversal of a hierarchy of elements defined in the schema:* (A schema of XML data (DTD) is reviewed, and handles are created, as similarly explained in column 4, lines 40-55. These handles are used to reference nodes for each element and/or attribute for direct access, since metadata for each node includes memory location information (column 6, lines 2-11). Furthermore, additional access parameters are generated from the schema, including hierarchical relationship access information as explained in column 9, lines 21-24. As shown in Fig. 5, hierarchical information includes the relationship of a *child node*/element *relative to a parent node*/element (such as the 'Price' element', with respect to the 'Item' element)).

However, Marcy does not expressly teach storing the XML data in a database in a storage system, *wherein one or more elements of the schema are stored in one or more respective columns of the database; identifying an element within the schema to associate with a named access procedure; wherein one or more elements of the schema are stored in one or more respective columns of the database*, determining if the element identified is appropriate for association with the named access procedure,

and if the element identified is appropriate for association, then creating the named access procedure, and associating the named access procedure with the element, the named access procedure providing direct access to the instance of the XML data.

Yet, Dreyband et al teaches *identifying an element within the schema to associate with a named access procedure* (Figure 3, paragraph 0029: whereas, elements defined in the xml file are identified, such as the element used in Dreyband's example xml file labeled: "name")*;* determining if the element identified is appropriate for association with the named access procedure (paragraph 0028: whereas, a check is performed to make sure that the xml document being used is "schema valid", thus inherently, the element defined in the xml document has been identified to be appropriate also), if *the element is identified for association, then creating the named access procedure, and associating the named access procedure with the element, the named access procedure providing access to the instance of the XML data* (paragraph 0028: whereas, a check is performed to make sure that the xml document being used is "schema valid", thus, the element defined in the xml document has been identified to be appropriate also. Additionally, as explained in paragraph 0029, and Figure 3: whereas, using the schema, an element is associated with a named access procedure by creating a java class named: "person", and having a named access procedure/attribute labeled "getName" (getName is associated with the child element labeled "name" in the provided schema shown in Figure2). Additionally, it is pointed out that Dreyband's named access procedure inherently teaches the use of access parameters as indicated in paragraph 0029: " getName return(s) data located in the name element". Thus, access parameter

information must have been known/used to know where to locate the data for the particular name element).

It would have been obvious to one of the ordinary skill in the art a the time of the invention to have modified Marcy's direct access node traversal method, such that the accessing of node data is accomplished through named access, as taught by Dreyband et al. The combination of Marcy and Dreyband et al would have allowed Marcy to have implemented "the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008)

However, although Marcy and Dreyband et al teach using named access to directly access node data, the combination of Marcy and Dreyband et al do not expressly teach *wherein one or more elements of the schema are stored in one or more respective columns of the database;*

Yet, Bhatt teaches *wherein one or more elements of the schema are stored in one or more respective columns of the database* (column 6, lines 30-37, column 10, lines 25-47, and column 12, lines 1-23: whereas, data is stored in columns of relational tables, and direct access is implemented by using link, path, and data offset data.).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy and Dreyband et al's method for using named access procedures to directly access a child node, such that the direct access comprises accessing a relational database for the child node, if an element is appropriate for association, as taught by Bhatt et al. The combination would have allowed Bhatt et al to

have "efficiently located a particular item, from a large group of similar items, by implementing a flexible, repeatable solution for efficiently extracting, storing and searching XML data" (Bhatt et al, column 3, lines 50-56).

With regards to claim 2, which depends on claim 1, for a method performing a similar method to claim 12, is rejected under the same rationale.

With regards to claim 3, which depends on claim 2, Marcy teaches a method comprising *access parameters includes offset information for each element*: Each element (for which the elements include child nodes, as shown in Figure 5) in a XML document (Figure 1) is assigned a memory offset location to be sent to an application for use as access parameters (column 6, lines 60-65).

With regards to claim 4, which depends on claim 1, Marcy does not teach a method which *the named access procedure is a procedure to get a value for the element or to set a value for the element.*

However, the combination of Marcy and Dreyband teaches a named access procedure (as similarly explained in the rejection for claim 1), and Dreyband et al further teaches a method *in which a named access procedure is defined to get a value for the child node or to set a value for the child node* (Figure 2 and 3: whereas, child elements disclosed in a schema shown in Figure 2 (such as child node ‘Name’), have

corresponding get and set functions shown in Figure 3 (such as ‘getName’ and ‘setName’)).

With regards to claim 8, which is dependent on claim 1, Marcy does not teach *an element is not appropriate for association if it is a node not defined in the schema*.

However, the combination of Marcy and Dreyband et al teaches the *association* (as similarly explained in the rejection for claim 1), and Dreyband et al further teaches the association further includes when *an element is not appropriate for association if it is a node that is not defined in the schema* (paragraph 0044: whereas, Dreyband et al’s system checks if the user xml document is schema valid). Dreyband et al’s system only makes the association if the user’s document is schema valid (paragraph 0029: whereas, a mapping happens when the user’s document is schema valid).

With regards to claim 10, which is dependent on claim 1, Marcy does not teach *the named access procedure is implemented as a bean accessor type.*

However, the combination of Marcy and Dreyband teaches *the named access procedure* (in claim 1, and is rejected under the same rationale), and Dreyband further teaches the named access procedure *is implemented as a bean accessor type* (Figure 2 and 3: whereas, child elements disclosed in a schema shown in Figure 2 (such as child node ‘Name’), have corresponding get and set functions shown in Figure 3 (such as ‘setName’)).

With regards to claim 11, Marcy teaches a method comprising:

- *Storing XML data in the database* (column 4, lines 1-29: whereas, XML data is stored in a document. The document is stored in computer memory/data-store/database)
- *Identifying an element associated with an instance of the XML data to access* (column 6, lines 23-33: whereas, an application program issues a request to access an element within an instance of XML data).
- *Determining using a processor at least one access parameter for the element relative to a second element in accordance with a schema, wherein the at least one access parameter allows the access procedure to have direct access to the element, ... without progressive traversal of a hierarchy of elements defined in the schema;* (A schema of XML data (DTD) is reviewed, and handles are created, as similarly explained in column 4, lines 40-55. These handles are used to reference nodes for each element and/or attribute for direct access, since metadata for each node includes memory location information (column 6, lines 2-11). Furthermore, additional access parameters are generated from the schema, including hierarchical relationship access information as explained in column 9, lines 21-24. As shown in Fig. 5, hierarchical information includes the relationship of a *child node*/element *relative to a parent node*/element (such as the ‘Price’ element’, with respect to the ‘Item’ element)).
- *If the element has not been associated with the named access procedure, then using a DOM API to access the element in the instance of the XML data* (column

2, lines 1-22: whereas, a DOM API is used to access element data through a search procedure)

However, Marcy does not teach ... *in a storage system, wherein one or more elements of a schema associated with the XML data are stored in one or more respective columns of the database;* determining if the element has been associated with *a named access procedure corresponding to the element; the named access procedure* to have direct access to *the element stored in a column of the database,* wherein direct access comprises *accessing the column of the database for the element to access an instance of the XML data associated with the element*.

Yet, Dreyband et al teaches … *determining if the element has been associated with a named access procedure corresponding to the element; the named access procedure to have access to the element stored in a database, wherein the access comprises accessing the database for the element to access an instance of the XML data associated with the element* (paragraph 0028: whereas, a check is performed to make sure that the xml document being used is "schema valid", thus, the element defined in the xml document has been identified to be appropriate also. Additionally, as explained in paragraph 0029, and Figure 3: whereas, using the schema, an element is associated with a named access procedure by creating a java class named: "person", and having a named access procedure/attribute labeled "getName" (getName is associated with the child element labeled "name" in the provided schema shown in Figure2). Additionally, it is pointed out that Dreyband's named access procedure inherently teaches the use of access parameters as indicated in paragraph 0029: " getName return(s) data located in

the name element". Thus, access parameter information must have been known/used to know where to locate the data for the particular name element).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's direct access node traversal method, such that the accessing of node data is accomplished through named access, as taught by Dreyband et al. The combination of Marcy and Dreyband et al would have allowed Marcy to have implemented "the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, although the combination of Marcy and Dreyband teach using named access to directly access node data, the combination of Marcy and Dreyband et al do not expressly teach in a storage system, *wherein one or more elements of a schema associated with the XML data are stored in one or more respective columns of the database.*

Yet, Bhatt et al teaches, *wherein one or more elements of a schema associated with the XML data are stored in one or more respective columns of the database, and* wherein direct access comprises *accessing the column of the database for the child node* to access an instance of the data (column 6, lines 30-37, column 10, lines 25-47, and column 12, lines 1-23: whereas, data is stored in columns of relational tables, and direct access is implemented by using link, path, and data offset data).

It would have been obvious to one of the ordinary skill in the art at the time of invention to have modified Marcy and Dreyband et al's method for using named access

procedures to directly access a child node, such that the direct access comprises accessing a relational database for the child node to directly access the child node, as taught by Bhatt et al. The combination would have allowed Bhatt et al to have "efficiently located a particular item, from a large group of similar items, by implementing a flexible, repeatable solution for efficiently extracting, storing and searching XML data" (Bhatt et al, column 3, lines 50-56).

With regards to claim 12, which depends on claim 11, Marcy teaches *a schema for XML data is known apriori* (column 4, lines 40-54: whereas, a XML parser receives a schema of XML data (DTD) for processing. However, Marcy does not explicitly teach *the named access procedure is based upon analysis of the schema*.
The combination of Marcy and Dreyband et al teaches a method *the named access procedure* (as similarly explained in the rejection for claim 1), and Dreyband et al further teaches the named access procedure *is based upon analysis of the schema* (Dreyband et al, paragraph 0029: whereas, a named access procedure/method is defined based on a valid schema, and the respective element defined in it).

With regards to claim 14, which depends on claim 11, Marcy teaches a method *in which other elements of the data not presently needed are not loaded into memory* (column 7, lines 10-20: whereas, values at child nodes are not loaded into memory, since the application program directly references a memory location in XML).

With regards to claim 18, Marcy teaches a method comprising:

- *Receiving the schema for the data that is based on the mark-up language* (column 4, lines 40-54: whereas, a XML parser receives a schema of XML data (DTD) for processing)
- *Storing the data in a database* (column 4, lines 1-29: whereas, XML data is stored in a document. The document is stored in computer memory/data-store/database)
- *Identifying a child node that is to be accessed within the data*: An application program is used to access content in an instance of XML data (column 6, lines 23-32: whereas, "the application program can obtain information and content on any part of the document"). Furthermore, it is inherent that the application program identifies a child node to be accessed within the data since the application access any part of the document, which includes child nodes of elements (column 5, lines 57-59).
- *Reviewing the schema to determine one or more access parameters relating to the child node:* A schema of XML data (DTD) is reviewed, and handles are created (column 4, lines 40-55:). These handles are used to reference nodes for each element and/or attribute for direct access, since metadata for each node includes memory location information (column 6, lines 2-11). Furthermore, additional access parameters are generated from the schema, including hierarchical relationship access information as explained in column 9, lines 21-24. As shown in Fig. 5, hierarchical information includes the relationship of a

child node/element *relative to a parent node*/element (such as the 'Price' element', with respect to the 'Item' element).

- *Using the one or more access parameters to access an instance of the data associated with the child node without progressive traversal of a hierarchy of nodes* (column 6, lines 23-32: whereas, an application program uses location information for a specified node (such as a child node), to directly reference the corresponding data in an XML instance).

However, although Marcy teaches access parameters being determined; Marcy does not expressly teach storing the data in a database *in a storage system, wherein the one or more child nodes of the data is stored in one or more respective columns of the database; wherein one or more elements in the XML data are designated as not eligible for the named access procedure,* and determining one or more access parameters for the child node relative to the parent node *in accordance with the schema, wherein* the at least one access parameter, *allows the name access procedure to have* direct *access to an instance of the child node in the data stored in a column of the database, wherein* direct access *comprises accessing the column of the database for the child node to access an instance of* the data.

Yet, Dreyband et al teaches *wherein* the at least one access parameter, *allows the name access procedure to have* access *to an instance of the child node in the data, wherein* access *comprises accessing the database for the child node to access an instance of* the data (Figure 3, paragraph 0029: whereas, using the schema, an element is associated with a named access procedure by creating a java class

named: “person”, and having a named access procedure/attribute labeled “getName” (getName is associated with the child element labeled “name” in the provided schema shown in Figure2). Additionally, it is pointed out that Dreyband’s named access procedure inherently teaches the use of access parameters as indicated in paragraph 0029: “ getName return(s) data located in the name element”. Thus, access parameter information must have been known/used to know where to locate the data for the particular name element).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's direct access node traversal method, such that the accessing of node data is accomplished through named access, as taught by Dreyband et al**.** The combination of Marcy and Dreyband et al would have allowed Marcy to have implemented “the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, although the combination of Marcy and Dreyband teach using named access to directly access node data, the combination of Marcy and Dreyband et al do not expressly teach the data is stored in a storage system, wherein the one or more child nodes of the data *is stored in one or more respective columns of the database,* and wherein direct access comprises *accessing the column of the database for the child node* to access an instance of the data.

Yet, Bhatt et al teaches the data is stored in a storage system, wherein the one or more child nodes of the data *is stored in one or more respective columns of the database,* and wherein direct access comprises *accessing the column of the database for the child node* to access an instance of the data (column 6, lines 30-37, column 10, lines 25-47, and column 12, lines 1-23: whereas, data is stored in columns of relational tables, and direct access is implemented by using link, path, and data offset data).

It would have been obvious to one of the ordinary skill in the art at the time of invention to have modified Marcy and Dreyband et al's method for using named access procedures to directly access a child node, such that the direct access comprises accessing a relational database for the child node to directly access the child node, as taught by Bhatt et al. The combination would have allowed Bhatt et al to have "efficiently located a particular item, from a large group of similar items, by implementing a flexible, repeatable solution for efficiently extracting, storing and searching XML data" (Bhatt et al, column 3, lines 50-56).

With regards to claim 19, Marcy teaches a method *in which the mark-up language is based on XML* (column 4, lines 40-54: whereas, XML markup language used).

With regards to claim 20, Marcy teaches a method *in which at least one access parameters is based on offset position* (column 6, lines 60-65: whereas, memory offset locations are used as access parameters).

With regards to claim 21, which depends on claim 18, Marcy does not explicitly teach method *in which a named access procedure is defined to get a value for the child node or to set a value for the child node*.

The combination of Marcy and Dreyband et al teaches *the named access procedure* (as similarly explained in the rejection for claim 1), and Dreyband et al further teaches the named access procedure *is defined to get a value for the child node or to set a value for the child node* (Figure 2 and 3: whereas, child elements disclosed in a schema shown in Figure 2 (such as child node ‘Name’), have corresponding get and set functions shown in Figure 3 (such as ‘getName’ and ‘setName’)).

With regards to claim 23, which depends on 18, Marcy teaches a method in which *the child node is not directly accessed if the node is not defined in the schema* (Fig. 3, column 5, lines 1-36: whereas, all elements, including child nodes that are not defined in the schema do not get referenced with an access handle. Since the user application (reference 6) does not receive the handle, the child node cannot be referenced/directly accessed).

With regards to claim 24, which depends on claim 18, Marcy teaches a method comprising *directly accessing a child node in which the method is performed* (in claim 18, and is rejected under the same rationale). Furthermore, Marcy teaches implementing the method using Java (column 8, lines 11-17), and thus it is inherent that

a method implemented in Java will *support the system's native data types* using the system's Java Virtual Machine.

With regards to claim 29, which depends on claim 18, Marcy teaches a method *in which other child nodes not presently needed are not loaded into memory* (column 7, lines 10-20: whereas, the child node data/values are not loaded into memory, since the application program references a memory location in XML instance.)

With regards to claim 30, Marcy teaches a system for:

- *Means for receiving a schema for the XML data*, as similarly explained in claim 18, and is rejected under the same rationale
- *A storage system for storing XML data in the database* (column 4, lines 1-29: whereas, XML data is stored in a document. The document is stored in computer memory/data-store/database)
- *A processor for determining at least one access parameter for the element relative to a second element in accordance with the schema, wherein the at least one access parameter allows* an *access procedure to have direct access to the element to access an instance of the XML data associated with the element without progressive traversal of a hierarchy of elements defined in the schema*, similarly explained in claim 18, and is rejected under the same rationale.

However, Marcy does not teach … *wherein one or more elements of the schema are stored in one or more respective columns of the database, means for identifying an*

element within the schema to associate with a named access procedure; Means for determining if the element identified is appropriate for association with the named access procedure, means for creating the named access procedure and associating the named access procedure with the element if the element is appropriate for association, the named access procedure providing direct access to the element within the XML data; and a named access procedure to have access to the element stored in a column of the database, wherein direct access comprises accessing the column of the database for the element.

Yet, Dreyband et al teaches means for *identifying an element within the schema to associate with a named access procedure; Means for determining if the element identified is appropriate for association with the named access procedure, means for creating the named access procedure and associating the named access procedure with the element if the element is appropriate for association, the named access procedure providing access to the element within the XML data* (paragraph 0028: whereas, a check is performed to make sure that the xml document being used is “schema valid“, thus, the element defined in the xml document has been identified to be appropriate also. Additionally, as explained in paragraph 0029, and Figure 3: whereas, using the schema, an element is associated with a named access procedure by creating a java class named: “person”, and having a named access procedure/attribute labeled “getName” (getName is associated with the child element labeled “name” in the provided schema shown in Figure2). Additionally, it is pointed out that Dreyband’s named access procedure inherently teaches the use of access parameters as indicated

in paragraph 0029: “ getName return(s) data located in the name element”. Thus, access parameter information must have been known/used to know where to locate the data for the particular name element).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's direct access node traversal method, such that the accessing of node data is accomplished through named access, as taught by Dreyband et al. The combination of Marcy and Dreyband et al would have allowed Marcy to have implemented “the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, although the combination of Marcy and Dreyband teach using named access to directly access node data, the combination of Marcy and Dreyband et al do not expressly teach in a storage system, *wherein one or more elements of the schema are stored in one or more respective columns of the database, an access procedure to have access to the element stored in a column of the database, wherein direct access comprises accessing the column of the database for the element*.

Yet, Bhatt et al teaches *wherein one or more elements of the schema are stored in one or more respective columns of the database, an access procedure to have access to the element stored in a column of the database, wherein direct access comprises accessing the column of the database for the element* (column 6, lines 30-37, column 10, lines 25-47, and column 12, lines 1-23: whereas, data is stored in columns of

relational tables, and direct access is implemented by using link, path, and data offset data).

It would have been obvious to one of the ordinary skill in the art at the time of invention to have modified Marcy and Dreyband et al's method for using named access procedures to directly access a child node, such that the direct access comprises accessing a relational database for the child node to directly access the child node, as taught by Bhatt et al. The combination would have allowed Bhatt et al to have "efficiently located a particular item, from a large group of similar items, by implementing a flexible, repeatable solution for efficiently extracting, storing and searching XML data" (Bhatt et al, column 3, lines 50-56).

With regards to claim 31 for a computer program product comprising a computer usable medium having executable code, is similar to a system performing a similar method to claim 30, and is rejected under the same rationale.

With regards to claim 32, for a system performing a method similar to claim 11, and is rejected under the same rationale.

With regards to claim 33, for a computer program product comprising a computer usable medium having executable code, is similar to a system performing a method of claim 11, and is rejected under the same rationale.

With regards to claim 34, for a system performing a method similar to claim 18, and is rejected under the same rationale.

With regards to claim 35, for a computer program product comprising a computer usable medium having executable code, performing a method similar to claim 18, and is rejected under the same rationale.

With regards to claim 36, which depends on claim 18, Marcy teaches *direct access is performed to a location in an XML document for the child node* (column 7, lines 9-30: whereas, various parsers are used to implement a direct access procedure using a location in an XML document for elements. Furthermore, the elements include child nodes, as shown in Figure 5.)

With regards to claim 37, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 1, is rejected under the same rationale.

With regards to claim 38, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 3, is rejected under the same rationale.

With regards to claim 40, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 8, is rejected under the same rationale.

With regards to claim 42, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 2, is rejected under the same rationale.

With regards to claim 43, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 3, is rejected under the same rationale.

With regards to claim 45, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 8, is rejected under the same rationale.

With regards to claim 47, which depends on claim 34, for a system performing a method that is similar to the method of claim 18, is rejected under the same rationale.

With regards to claim 48, which depends on claim 34, for a system performing a method that is similar to the method of claim 20, is rejected under the same rationale.

With regards to claim 50, which depends on claim 34, for a system performing a method that is similar to the method of claim 24, is rejected under the same rationale.

With regards to claim 51, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 18, is rejected under the same rationale.

With regards to claim 52, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 20, is rejected under the same rationale.

With regards to claim 54, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 24, is rejected under the same rationale.

With regards to claim 56, which depends on claim 32, for a system performing a method similar to the method of claim 14, is rejected under the same rationale.

With regards to claim 60, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 14, is rejected under the same rationale.

5. Claims 15-17, 25-28, 57, 58, 61, and 62 are rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy (US Patent: 6,662,342 B1, published: Dec. 9, 2003, filed: Dec. 13, 1999), in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Bhatt et al (US Patent: 6,799,184 B2, issued: Sep. 28, 2004, filed: Jan. 30, 2002), and further in view of Wan (US Patent: 2003/0233618 A1, published: Dec. 18, 2003, filed: Jun. 16, 2003, Foreign priority: Jun. 17, 2002).

With regards to claim 15, which depends on claim 11, Marcy does not teach a method in which *the element is at a known offset from a parent location*.

However, Wan teaches *the element is at a known offset from a parent location* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes using offsets based off of the memory location of a parent node).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's application for accessing node data to further include the method for accessing child nodes via offsets based off of a location of the corresponding parent nodes as taught by Wan. The combination of Marcy, Dreyband et al, Bhatt et al, and Wan would have allowed Marcy's system to have been able to have reduced iterative traversal of an XML instance.

With regards to claim 16, which depends on claim 15, Marcy teaches a method *in which the known offset is managed independently of the XML data* (column 6, lies 60-65, Figure 3: whereas, an application program (reference number 6), receives offset information from the XML parser, and thus, is managed independently of the data).

With regards to claim 17, which depends on claim 11, Marcy does not teach a method *in which a memory layout associated with the XML data is maintained as a flat layout.*

However, Wan teaches a method *in which a memory layout associated with the XML data is maintained as a flat layout* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes in a flat layout (index/list)).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's application for accessing node data to further include the method for accessing the memory location of nodes in a flat layout. The combination of Marcy, Dreyband et al, Bhatt et al, and Wan would have allowed Marcy's system to have been able to improve the usage of memory.

With regards to claim 25, which depends on claim 18, Marcy does not teach a method *in which direct access is performed to an offset location for the child node*.

However, Wan teaches a method *in which direct access is performed to an offset location for the child node* (Table B and D: whereas, Table B represents an XML

instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes using offsets).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's application for accessing node data, to further include the method for accessing child nodes via offsets as taught by Wan. The combination of Marcy, Dreyband et al, Bhatt et al, and Wan would have allowed Marcy system to have parsed "schemas of structural documents to determine predefined deterministic relationships between components of structured documents to be indexed" (paragraph 0009).

With regards to claim 26, which depends on claim 25, Marcy does not teach a method *in which the child node is at a known offset from a location for the parent node*.

However, Wan teaches a method *in which the child node is at a known offset from a location for the parent node* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes using offsets based off of the memory location of a parent node).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's application for accessing node data to further include the method for accessing child nodes via offsets based off of a location of the corresponding parent nodes as taught by Wan. The combination of Marcy, Dreyband et

al, Bhatt et al, and Wan would have allowed Marcy's system to have been able to have reduced iterative traversal of an XML instance.

With regards to claim 27, which depends on claim 26, Marcy teaches a method *in which a mapping of the known offset is managed independently of the data* (column 6, lies 60-65, Figure 3: whereas, an application program (reference number 6), receives offset information from the XML parser, and thus, is managed independently of the data).

With regards to claim 28, which depends on claim 25, Marcy does not teach a method *in which memory layout associated with the data is maintained as a flat layout*.

However, Wan teaches a method *in which memory layout associated with the data is maintained as a flat layout* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes in a flat layout (index/list)).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's application for accessing node data to further include the method for accessing the memory location of nodes in a flat layout. The combination of Marcy, Dreyband et al, Bhatt et al, and Wan would have allowed Marcy's system to have been able to improve the usage of memory.

With regards to claim 57, which depends on claim 32, for a system performing a method similar to the method of claim 15, is rejected under the same rationale.

With regards to claim 58, which depends on claim 32, for a system performing a method similar to the method of claim 17, is rejected under the same rationale.

With regards to claim 61, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 15, is rejected under the same rationale.

With regards to claim 62, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 17, is rejected under the same rationale.

6. Claims 5, 6, 7, 9, 13, 22, 39, 41, 44, 46, 49, 53, 55, and 59 are rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy (US Patent: 6,662,342 B1, published: Dec. 9, 2003, filed: Dec. 13, 1999), in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Bhatt et al (US Patent: 6,799,184 B2, issued: Sep. 28, 2004, filed: Jan. 30, 2002), and further in view of JAXB (Sun Microsystems, pages 58, and 74, published: January 8, 2003).

With regards to claim 5, which depends on claim 1, Marcy does not explicitly teach a method *in which direct mapping is performed to an intended data type for the element.*

However, JAXB teaches a method *in which direct mapping is performed to an intended data type for an element* (page 74: whereas, based on the data type disclosed in a schema, a mapping is performed to an intended Java data type, as shown in the example, an 'int' datatype in the schema is mapped to an 'int' Java datatype, and a 'float' datatype in the schema is mapped to a 'float' Java datatype).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's system to include a method of directly mapping datatypes as taught by JAXB. The combination of Marcy, Dreyband et al, Bhatt et al, and JAXB would have allowed an application that utilized Marcy's system, to have accessed data more efficiently, without the overhead of any additional datatype mapping steps.

With regards to claim 6, which depends on claim 5, Marcy does not explicitly teach a method *in which a conversion to a string datatype is not performed when mapping to the intended datatype.*

However, the combination of Marcy, Dreyband et al, Bhatt et al, and JAXB teaches *the conversion* (as similarly explained in the rejection for claim 5), and JAXB further teaches the conversion further includes a method in *which a conversion to a*

string datatype is not performed when mapping to the intended datatype (page 74: whereas, an 'int' mapping is performed).

With regards to claim 7, which depends on claim 5, Marcy does not explicitly teach a method *in which the mapping is a close-matching datatype.*

However, JAXB teaches a method in *which the mapping is a close matching datatype* (page 58: whereas, mapping to a javatype is based on the defined schema datatype. As shown, an 'unsigned int' datatype in a schema is mapped to the closest matching datatype in Java, which is a 'long').

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's XML access system to further provide a closest matching datatype that is supported for a particular system/application. The combination of Marcy, Dreyband et al, Bhatt et al, and JAXB would have allowed Marcy's system operate flexibly on various different platforms using different datatypes.

With regards to claim 9, which depends on claim 1, Marcy does not explicitly teach a method *in which the element is appropriate for association if it corresponds to a native datatype of the system in which the method is performed* (page 58, Table 5-1: whereas, all the schema datatypes disclosed are mapped to all supported Java datatypes. Furthermore, it is inherent that Java datatypes will be supported by the native datatypes available in the system through the use of the platform specific Java Virtual Machine).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy's system to further include a method for associating elements that map to Java datatypes for system independent datatype support. The combination of Marcy, Dreyband et al, Bhatt et al, and JAXB would have allowed Marcy's system to have operated independent of platform type.

With regards to claim 13, which depends on claim 11, for a method performing a similar method to claim 5, is rejected under the same rationale.

With regards to claim 22, which depends on claim 18, Marcy teaches a method comprising *identifying a child node* that is to be accessed within the data: An application program is used to access content in an instance of XML data (column 6, lines 23-32: whereas, "the application program can obtain information and content on any part of the document"). Furthermore, it is inherent that the application program identifies a child node to be accessed within the data since the application access any part of the document, which includes child nodes of elements (column 5, lines 57-59). However, Marcy does not explicitly teach a method *in which direct mapping is performed to an intended datatype for the* child node.

JAXB teaches a method *in which direct mapping is performed to an intended datatype for a child node/*element (page 74: whereas, based on the datatype disclosed in a schema, a mapping is performed to an intended Java datatype, as shown in the

example, an ‘int’ datatype in the schema is mapped to an ‘int’ Java datatype, and a ‘float’ datatype in the schema is mapped to a ‘float’ Java datatype).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Marcy’s method of accessing child node data to further include a method of direct mapping to an intended datatype as taught by JAXB. The combination Marcy, Dreyband et al, Bhatt et al, and JAXB would have allowed Marcy’s system to have performed faster child node data retrieval.

With regards to claim 39, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 5, is rejected under the same rationale.

With regards to claim 41, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 9, is rejected under the same rationale.

With regards to claim 44, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 5, is rejected under the same rationale.

With regards to claim 46, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 9, is rejected under the same rationale.

With regards to claim 49, which depends on claim 34, for a system performing a method that is similar to the method of claim 22, is rejected under the same rationale.

With regards to claim 53, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 22, is rejected under the same rationale.

With regards to claim 55, which depends on claim 32, for a system performing a method similar to the method of claim 5, is rejected under the same rationale.

With regards to claim 59, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 5, is rejected under the same rationale.

Response to Arguments

7. Applicant's arguments with respect to claims 1-62 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to WILSON TSUI whose telephone number is (571)272-7596. The examiner can normally be reached on Monday - Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Stephen Hong can be reached on (571) 272-4124. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/CESAR B PAULA/
Primary Examiner, Art Unit 2178

/Wilson Tsui/
Patent Examiner
Art Unit: 2178
May 19, 2009